FORM PTO-1390 (Modified) (REV 11-98) U.S. DEPARTMENT OF COMMERCE PATENT AND TRADEMARK OFFICE

TRANSMITTAL LETTER TO THE UNITED STATES DESIGNATED/ELECTED OFFICE (DO/EO/US) CONCERNING A FILING UNDER 35 U.S.C. 371

ATTORNEY'S DOCKET NUMBER: **27428/36596**

U.S. APPLICATION NO (IF KNOWN, SEE 37 CFR): **09/601167**

INTERNATIONAL APPLICATION NO	INTERNATIONAL FILING DATE	PRIORITY DATE CLAIMED
PCT/DE99/00213	**28 July 1999**	**31 January 1998**

TITLE OF INVENTION
COMPRESSION OF DATA WITH SYNTACTIC STRUCTURE

APPLICANT(S) FOR DO/EO/US
PETER ECK,, ROLF MATZNER and CHANGSONG XIE

Applicant herewith submits to the United States Designated/Elected Office (DO/EO/US) the following items and other information.

1. ☒ This is a **FIRST** submission of items concerning a filing under 35 U.S.C. 371.
2. ☐ This is a **SECOND** or **SUBSEQUENT** submission of items concerning a filing under 35 U.S.C. 371.
3. ☐ This is an express request to begin national examination procedures (35 U.S.C. 371(f)) at any time rather than delay examination until the expiration of the applicable time limit set in 35 U.S.C. 371(b) and PCT Articles 22 and 39(1).
4. ☒ A proper Demand for International Preliminary Examination was made by the 19th month from the earliest claimed priority date.
5. ☒ A copy of the International Application as filed (35 U.S.C. 371 (c) (2))
   a. ☐ is transmitted herewith (required only if not transmitted by the International Bureau).
   b. ☒ has been transmitted by the International Bureau.
   c. ☐ is not required, as the application was filed in the United States Receiving Office (RO/US).
6. ☒ A translation of the International Application into English (35 U.S.C. 371(c)(2)).
7. ☐ A copy of the International Search Report (PCT/ISA/210).
8. ☒ Amendments to the claims of the International Application under PCT Article 19 (35 U.S.C. 371 (c)(3))
   a. ☐ are transmitted herewith (required only if not transmitted by the International Bureau).
   b. ☐ have been transmitted by the International Bureau.
   c. ☐ have not been made; however, the time limit for making such amendments has NOT expired.
   d. ☒ have not been made and will not be made.
9. ☐ A translation of the amendments to the claims under PCT Article 19 (35 U.S.C. 371(c)(3))
10. ☐ An oath or declaration of the inventor(s) (35 U.S.C. 371 (c)(4)).
11. ☐ A copy of the International Preliminary Examination Report (PCT/IPEA/409).
12. ☐ A translation of the annexes to the International Preliminary Examination Report under PCT Article 36 (35 U.S.C. 371 (c)(5)).

Items 13 to 20 below concern document(s) or information included:

13. ☐ An Information Disclosure Statement under 37 CFR 1.97 and 1.98.
14. ☐ An assignment document for recording. A separate cover sheet in compliance with 37 CFR 3.28 and 3.31 is included.
15. ☒ A **FIRST** preliminary amendment.
16. ☐ A **SECOND** or **SUBSEQUENT** preliminary amendment.
17. ☒ A substitute specification.
18. ☐ A change of power of attorney and/or address letter.
19. ☒ Certificate of Mailing by Express Mail
20. ☐ Other items or information:

534 Rec'd PCT/PTO 28 JUL2000

U.S. APPLICATION NO (IF KNOWN, SEE 37 CFR)	INTERNATIONAL APPLICATION NO	ATTORNEY'S DOCKET NUMBER
09/601167	PCT/DE99/00213	27428/36596

21. The following fees are submitted .

BASIC NATIONAL FEE (37 CFR 1.492 (a) (1) - (5)) :

		CALCULATIONS PTO USE ONLY
☐	Neither international preliminary examination fee (37 CFR 1.482) nor international search fee (37 CFR 1 445(a)(2) paid to USPTO and International Search Report not prepared by the EPO or JPO	$970.00
☒	International preliminary examination fee (37 CFR 1.482) not paid to USPTO but Internation Search Report prepared by the EPO or JPO	$840.00
☐	International preliminary examination fee (37 CFR 1.482) not paid to USPTO but international search fee (37 CFR 1 445(a)(2)) paid to USPTO	$690.00
☐	International preliminary examination fee paid to USPTO (37 CFR 1.482) but all claims did not satisfy provisions of PCT Article 33(1)-(4)	$670.00
☐	International preliminary examination fee paid to USPTO (37 CFR 1.482) and all claims satisfied provisions of PCT Article 33(1)-(4)	$96.00

ENTER APPROPRIATE BASIC FEE AMOUNT =	**$840.00**	
Surcharge of **$130.00** for furnishing the oath or declaration later than ☐ 20 ☐ 30 months from the earliest claimed priority date (37 CFR 1.492 (e)).	**$0.00**	

CLAIMS	NUMBER FILED	NUMBER EXTRA	RATE		
Total claims	7 - 20 =	0	x $18.00	**$0.00**	
Independent claims	2 - 3 =	0	x $78.00	**$0.00**	
Multiple Dependent Claims **(check if applicable)**.			☐	**$0.00**	
TOTAL OF ABOVE CALCULATIONS =				**$840.00**	
Reduction of 1/2 for filing by small entity, if applicable. Verified Small Entity Statement must also be filed (Note 37 CFR 1.9, 1.27, 1 28) **(check if applicable)**.			☐	**$0.00**	
SUBTOTAL =				**$840.00**	
Processing fee of **$130.00** for furnishing the English translation later than ☐ 20 ☐ 30 months from the earliest claimed priority date (37 CFR 1.492 (f)).			+	**$0.00**	
TOTAL NATIONAL FEE =				**$840.00**	
Fee for recording the enclosed assignment (37 CFR 1.21(h)). The assignment must be accompanied by an appropriate cover sheet (37 CFR 3.28, 3.31) **(check if applicable).**			☐	**$0.00**	
TOTAL FEES ENCLOSED =				**$840.00**	
				Amount to be: refunded	$
				charged	$

☒ A check in the amount of **$840.00** to cover the above fees is enclosed.

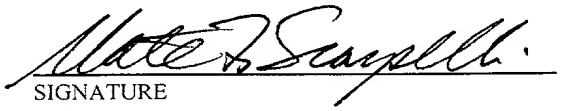
☐ Please charge my Deposit Account No. in the amount of to cover the above fees.
A duplicate copy of this sheet is enclosed.

☒ The Commissioner is hereby authorized to charge any fees which may be required, or credit any overpayment to Deposit Account No. **13-2855** A duplicate copy of this sheet is enclosed.

NOTE: Where an appropriate time limit under 37 CFR 1.494 or 1.495 has not been met, a petition to revive (37 CFR 1.137(a) or (b)) must be filed and granted to restore the application to pending status.

SEND ALL CORRESPONDENCE TO:

Nate F. Scarpelli
MARSHALL, O'TOOLE, GERSTEIN,
MURRAY & BORUN
233 South Wacker Drive
6300 Sears Tower
Chicago, Illinois 60606-6402
Tel.: (312) 474-6300
Fax: (312) 474-0448



SIGNATURE

Nate F. Scarpelli
NAME

22,320
REGISTRATION NUMBER

July 28, 2000
DATE

PATENT

Attorney's Docket No: 27428/36596

Applicant or Patentee: PETER ECK, ET AL.

Serial or Patent No: 09/601,167

Filed or Issued: July 28, 2000

For: COMPRESSION OF DATA WITH SYNTACTIC STRUCTURE

VERIFIED STATEMENT (DECLARATION) CLAIMING SMALL ENTITY STATUS (37 CFR 1.9(f) and 1.27(c)) -- SMALL BUSINESS CONCERN

I hereby declare that I am

☐ The owner of the small business concern identified below:

☒ An official of the small business concern empowered to act on behalf of the concern identified below:

NAME OF CONCERN Syntion AG

ADDRESS OF BUSINESS Leonrodplatz 2, 80636 Munich, Federal Republic of Germany

I hereby declare that the above-identified small business concern qualifies as a small business concern as defined in 13 CFR 121.12, and reproduced in 37 CFR 1.9(d), for purposes of paying reduced fees under Section 41(a) and (b) of Title 35, United States Code, in that the number of employees of the concern, including those of its affiliates, does not exceed 500 persons. For purposes of this statement, (1) the number of employees of the business concern is the average over the previous fiscal year of the concern of the persons employed on a full-time, part-time or temporary basis during each of the pay periods of the fiscal year, and (2) concerns are affiliates of each other when either, directly or indirectly, one concern controls or has the power to control the other, or a third-party or parties controls or has the power to control both.

I hereby declare that rights under contract or law have been conveyed to, and remain with, the small business concern identified above with regard to the invention, entitled COMPRESSION OF DATA WITH SYNTACTIC STRUCTURE, by inventor(s) Peter Eck, Rolf Matzner, and Changsong Xie,

described in

☐ The specification filed herewith.

☒ Application Serial No. 09/601,167, filed July 28, 2000.

☐ Patent No. _____, issued _____.

If the rights held by the above-identified small business concern are not exclusive, each individual, concern or organization having rights to the invention is listed below* and no rights to the invention are held by any person, other than the inventor, who would not qualify as an independent inventor under 37 CFR 1.9(c), if that person made the invention, or by any concern which would not qualify as a small business concern under 37 CFR 1.9(d) or a nonprofit organization under 37 CFR 1.9(e).

*NOTE: *Separate verified statements are required from each named person, concern or organization having rights to the invention averring to their status as small entities. (37 CFR 1.27).*

NAME: _____
ADDRESS: _____
☐ INDIVIDUAL ☐ SMALL BUSINESS CONCERN ☐ NONPROFIT ORGANIZATION

NAME: _____
ADDRESS: _____
☐ INDIVIDUAL ☐ SMALL BUSINESS CONCERN ☐ NONPROFIT ORGANIZATION

I acknowledge the duty to file, in this application or patent, notification of any change in status resulting in loss of entitlement to small entity status prior to paying, or at the time of paying, the earliest of the issue fee or any maintenance fee due after the date on which status as a small entity is no longer appropriate. (37 CFR 1.28(b)).

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under Section 1001 of Title 18 of the United States Code, and that such willful false statements may jeopardize the validity of the application, any patent issuing thereon, or any patent to which this verified statement is directed.

NAME OF PERSON SIGNING: ROLF MATZNER

TITLE OF PERSON OTHER THAN OWNER: CHIEF TECHNOLOGY OFFICER

ADDRESS OF PERSON SIGNING: JOSEF-FRANKL-STR. 9A
80995 MUNICH, GERMANY

SIGNATURE: _____ Date 28-AUG-2000

PATENT

27428/36596

IN THE UNITED STATES
PATENT AND TRADEMARK OFFICE

APPLICANT:	"EXPRESS MAIL" mailing label No. EL 566462735 US.
PETER ECK, ET AL.	Date of Deposit: July 28, 2000
SERIAL NO.: To Be Determined	I hereby certify that this paper (or fee) is being
FILED: Filed Herewith	deposited with the United States Postal Service
FOR: COMPRESSION OF DATA WITH SYNTACTIC STRUCTURE	"EXPRESS MAIL POST OFFICE TO ADDRESSEE" service under 37 CFR §1.10 on the date
Based on: PCT/DE99/00213 Filed: 31 July 1997 and based on 19803845.3 filed 31 January 1998	indicated above and is addressed to: Assistant Commissioner for Patents, Washington, D.C. 20231
GROUP ART UNIT: To Be Determined	Laura Frasher Laura Frasher
EXAMINER: To Be Determined	

PRELIMINARY AMENDMENT "A" ACCOMPANYING
SECTION 371 NATIONAL STAGE APPLICATION AS FILED

Assistant Commissioner
for Patents
BOX PCT
Washington, D.C. 20231

Sir:

Please amend the accompanying Section 371 national stage application as filed as follows:

IN THE SPECIFICATION:

In place of the enclosed literal translation of the specification of the international application, marked

"APPLICATION WITH LITERAL TRANSLATION", please substitute the enclosed specification marked "SUBSTITUTE SPECIFICATION".

The substitute specification does not include any new matter. Amendments made to the specification text are as follows:

On page 3, line 20 "4.1 Summary of the Invention" inserted.

On page 3, lines 22-32 and page 4, lines 1-3 a statement equivalent to Claim 1 has been inserted.

On page 4, lines 5-27 a statement equivalent to Claim 8 has been inserted.

On page 5, lines 2-22 a statement equivalent to Claim 9 has been inserted.

On page 10, line 11 "6.3 Brief Description of the Drawings" inserted.

On page 11, line 18 "6.3" change to "6.4".

On page 12, line 21 "6.4" changed to "6.5".

On page 13, line 11 "6.5" changed to "6.6".

On page 13, line 20 "Claim 22" corrected to "these claims".

IN THE CLAIMS:

Please cancel claims 1-10 without prejudice.

Please add the following claims 11-17:

--11. A method of transmitting a text represented by digital data, the structure of which is defined by a grammar with grammatical rules, from a transmitter to a receiver comprising the steps of:

(a) in the transmitter, for the purpose of encoding, the grammatical rules contained in the text are converted to a sequence of syntax-directed coding (SDC) symbols by parsing the text and generating a parse tree, such that to each node in this parse tree there is attributed an SDC symbol, which out of all the rules permitted by the grammar at this site unambiguously identifies the rule that is actually contained in the text, and such that the SDC symbols are concatenated, according to a fixed order of traversing all nodes of the parse tree, to form a linear sequence of SDC symbols;

(b) sending the SDC symbol sequence from a transmitter to a receiver for storage in the receiver;

(c) in the receiver, for the purpose of decoding, in a stack machine the grammatical rule corresponding to each stored SDC symbol in the sequence of SDC symbols is executed in order to generate output data that contain the text, in which process an uppermost entry in a stack memory of the stack machine is replaced according to a grammar production determined by an SDC symbol that has been input, and parts of the grammatical rules that cannot yet be completely processed are deposited in a stack memory, whereas

parts of the production for which substitution is complete are processed immediately to form part of an executable program for a real processor or a virtual machine.--

--12. A method as claimed in Claim 11, including the steps of:

(a) initializing the stack machine by depositing a specific start symbol or non-terminal symbol into an empty stack memory;

(b) reading an uppermost symbol from the stack memory;

(c) testing whether the read symbol is a terminal or a non-terminal symbol;

(d_1) if the read symbol is a terminal symbol, outputting the symbol regardless of whether additional symbols are present in the stack memory, and continuing the method with step (b) above and terminating the method whenever the stack memory is empty, or

(d_2) if the read symbol is a non-terminal symbol, reading the next SDC symbol from the input stream;

(e) dependent on what SDC symbol has been read, selecting only one alternative or chain of terminal and/or non-terminal symbols out of the set of alternatively applicable replacement rules or productions that are valid for the non-terminal symbol currently being processed;

(f) placing the chain of terminal and/or non-terminal symbols into the stack memory and then continuing the method with step (b) above.--

--13. A method as claimed in Claim 11, wherein

(a) to each node in the parse tree there is attributed an SDC symbol and the probability distribution of all the SDC symbols possible in this node,

(b) the sequence of SDC symbols is subjected to entropy encoding in conformity with the associated probability distributions, and

(c) the entropy decoding is carried out with a probability distribution of SDC symbols identical to that used for the entropy encoding.--

--14. A method as claimed in Claim 13, wherein

(a) the probability distribution of the rules that can be applied in a node, starting from an initial distribution, is adapted at each appearance of an SDC symbol in such a way that the probability of the SDC symbol that appears is increased and the probability of all other symbols is correspondingly reduced,

(b) the currently valid distribution of occurrence probabilities is assigned to the SDC symbols of the associated node type,

(c) the probability distribution of all SDC symbols in the current node, together with the SDC symbol to be encoded, forms a model for an arithmetic encoding,

(d) during decoding an end of the text is recognized by the fact that the stack memory is empty, and

(e) an End-Of-Message (EOM) symbol required for arithmetic coding is eliminated.--

--15. Apparatus for transmitting a text represented by digital data, the structure of which is defined by a grammar with grammatical rules, from a transmitter to a receiver, comprising

(a) an encoder, which comprises

(aa) a scanner to transform text consisting of a sequence of readable characters into a sequence of terminal symbols,

(ab) a parser to find grammatical rules, the successive application of which was originally used to generate the sequence of terminal symbols,

(ac) a mapper, which unambiguously associates syntax-directed symbols with the rules identified by the parser and outputs these symbols in a fixed sequence, and

(b) a decoder, which comprises

(ba) a stack machine that, according to an uppermost symbol in the stack memory and any adjacent SDC symbol, outputs the already fixed terminal symbol, or deposits

the sequence of terminal and/or non-terminal symbols associated with the current symbol into the stack memory, and

(bb) a lexicon that replaces the terminal symbols by chains of readable alphanumeric characters, such that the decoder immediately processes parts of the production for which substitution is complete to form part of an executable program for a real processor or a virtual machine.--

--16. Apparatus as claimed in Claim 15, in which the text transformed by the scanner into a sequence of terminal symbols is a program present in the source text or in a form derived from the source text by a preprocessor, and in which a code generator is provided that generates from the sequence of terminal symbols an executable machine code or an intermediate code to be executed on a virtual machine.--

--17. Apparatus as claimed in Claim 15, which comprises

(a) in the transmitter

(aa) a table that contains the probability distributions of the SDC symbols for each node type in a parse tree generated by a parser and the contents of which are established at initialization with fixed initial probability distributions for each node type,

(ab) an adapter that updates the probability distribution of the SDC symbols for a node type valid at the current moment with reference to the existing probability distribution, the SDC symbol to be encoded and the current node type, and enters this new probability distribution into the table, and

(ac) an arithmetic encoder that encodes the SDC symbol currently to be encoded with the currently valid probability distribution supplied by the adapter;

(b) in the receiver

(ba) a table that contains the probability distributions of the SDC symbols for each node type, the contents of which are established at initialization with fixed initial probability distributions for each node type,

(bb) an adapter that updates the probability distribution of the SDC symbols for the node type valid at the current moment, as established by the stack machine, with reference to the existing probability distribution, the SDC symbol to be decoded and the current node type, and enters this new probability distribution into the table, and

(bc) an arithmetic decoder which, with reference to the currently valid probability distribution of the current node type supplied by the adapter, decodes the next SDC symbol and sends it to the stack machine for further processing.--

REMARKS

The enclosed translation of the international application marked "APPLICATION WITH LITERAL TRANSLATION" includes the translated specification, claims, abstract, and drawings text.

The substitute specification marked "SUBSTITUTE SPECIFICATION" includes a revised specification and abstract for use in the examination of this application and does not include any new matter with respect to the literal translation.

It is requested that the Examiner uses the "SUBSTITUTE SPECIFICATION" in place of the literal translation during the examination of this application.

Respectfully submitted,

Nate F. Scarpelli, Reg. #22,320
MARSHALL, O'TOOLE, GERSTEIN,
MURRAY & BORUN
6300 Sears Tower
233 S. Wacker Drive
Chicago, Illinois 60606
Tel.: (312) 474-6300
Fax: (312) 474-0448

Dated: July 28, 2000

27428/36596

Please use this revised translation for examination of 371 case - PCT/DE 99/00213

"SUBSTITUTE SPECIFICATION"

COMPRESSION OF DATA WITH SYNTACTIC STRUCTURE

1. Introduction to the General Topic

The coding of source files has important technical applications in the area of transmission and storage of information. The desire to have a low data rate in the transmission of information, and a small space requirement in the storage of information, has given rise to the demand for a form of information coding that is as brief as possible, i.e. one with little or, ideally, no redundancy.

2. State of the Art

2.1. Forms of Redundancy

Redundancy in streams of source symbols takes two basic forms: it can appear as a differential frequency of occurrence of the elements in the set of source symbols, and in the form of statistical dependencies between temporally separated source symbols, which usually are caused by source-specific *formation laws* for the construction of a text as a sequence of source symbols.

3. Disadvantages of Known Methods

For the compression of messages from sources that suffer from redundancy and emit source symbols that are not uniformly distributed but are statistically independent, a procedure according to Huffman [1] is known. Being specially intended for this type of source, the

procedure enables maximal compression only under the extremely favorable condition that the statistical properties of the source are known or can be reliably estimated.

Whereas estimation of the probabilities of occurrence of individual source symbols, which is necessary for Huffman coding, can be done with sufficient accuracy even in the case of moderately long texts, the estimation of statistical dependencies between temporally separate source symbols as a rule requires extensive observation of the source, which goes far beyond what is feasible both technically (storage capacity) and in terms of observation time. For this reason the known or similar procedures always involve making assumptions about the underlying formation law or crucial characteristics thereof, and then of course are suitable only for sources with this formation law. This qualification also applies to the Lempel-Ziv (LZ) procedure for data compression [2], which has been known since the end of the 1970's. This procedure is suitable for the compression of arbitrary linear symbol sequences; that is, it can be applied with no knowledge of the statistical properties of the information source. The main areas of application today are the compression of pictorial data (GIF file format) and arbitrary files (archiving programs gnuzip, pkzip). In contrast to Huffman coding, this procedure utilizes statistical dependencies between temporally successive symbols, by taking into account repeatedly occurring subsequences of symbols (strings). Hence LZ compression is suitable only for sources, the formation law of which leads to the

frequent occurrence of particular strings. The redundancy introduced into a text by following grammatical rules during its construction is not accessible to the known compression procedures because of the recursive structure of grammatical rules, since these procedures merely employ symbol frequencies or repetitive chains of alphanumeric characters for compression.

4. Definition of the Object of the Invention

The object of the present invention is to provide a method and apparatus for implementing the method that enable a text consisting of a stream of alphanumeric characters, and is constructed according to grammatical rules, to be encoded in such a way as to remove the redundancy produced as a result of the limitation of all possible character sequences by the grammar.

4.1 Summary of the Invention

According to a first aspect of the present invention there is provided a method for the compressed transmission and/or storage of a text represented by digital data, the structure of which is defined by a grammar, wherein

prior to storage and/or transmission the text is converted into a linear sequence of symbols that specify the successive application of grammatical rules to form the text, and

to decode the text after the storage and/or transmission, for every symbol thus produced by syntax-

directed coding (SDC) the grammatical rule corresponding to that particular SDC symbol is performed and particular output data are generated by this rule.

According to a second aspect of the present invention there is provided an apparatus for the compressed transmission and/or storage of a text represented by digital data, the structure of which is defined by a grammar, comprising

a scanner to transform the text comprising a sequence of readable characters into a sequence of terminal symbols,

a parser to find the grammatical rules, the successive application of which was originally used to generate the sequence of terminal symbols,

a mapper, which unambiguously associates syntax-directed symbols with the rules identified by the parser and outputs these symbols in a fixed sequence, and

a decoder comprising

(a) a stack machine which, according to the uppermost symbol in the stack memory and any adjacent SDC symbol, outputs the already fixed terminal symbol, or deposits the sequence of terminal and/or non-terminal symbols associated with the current symbol into the stack memory, and

(b) a lexicon that replaces the terminal symbols by chains of readable alphanumeric characters.

According to a third aspect of the present invention there is provided an encoder comprising

a scanner to transform a program, present in the source text or in a form derived from the source text by a preprocessor, into a sequence of terminal symbols,

a parser to find the grammatical rules, the successive application of which was originally used to generate the sequence of terminal symbols,

a mapper, which unambiguously associates syntax-directed symbols with the rules identified by the parser and outputs these symbols in a fixed sequence,

and a decoder comprising

a stack machine which, according to the uppermost symbol in the stack memory and any associated SDC symbol, outputs the already fixed terminal symbol, or deposits the sequence of terminal and/or non-terminal symbols associated with the current symbol into the stack memory, and

a code generator that generates from the sequence of terminal symbols executable machine code, or intermediate code to be executed on a virtual machine.

5. Advantages of the Solution in Accordance with the Invention

With the solution in accordance with the invention, enabling data reduction in dependence on the programming-language syntax employed, it now becomes possible in an especially advantageous manner to remove, asymptotically almost completely, redundancy produced in

texts during their construction because of the need to follow certain grammatical rules.

The invention has proved this by analytical calculation of the capacity of its data stream, encoded in a syntax-directed manner, using the example of a language with characteristics typical of programming languages (mathematical expressions, if-then-else construct, etc.). In this process it was shown that an appreciable amount of the redundancy that conventional methods (Lempel-Ziv, Huffman) are designed to address has not previously been eliminated by these methods.

The new method of syntax-directed coding, in contrast, is capable of reducing the amount of data by an additional *ca.* one-half in comparison, for instance, to a result obtained with the Lempel-Ziv compression procedure.

Thus the method in accordance with the invention, together with the apparatus specified for its implementation, is predestined for all applications that involve the transmission or storage of syntactically structured texts. Examples include the transmission of Java applets in the Internet or an intranet, the transmission or storage of Postscript files, the transmission and storage of MPEG4-encoded video data, and the use of higher protocols in communication. Especially for the transmission of programs, such as Java applets, syntax-directed coding (SDC) is particularly suitable because the parser is a component of both a compiler and an SDC coder, so that SDC can be

organically integrated into the data flow from the writing to the running of the program.

6. Mode of operation of the invention

6.1 Syntactically structured source

The invention takes as a point of departure a formal language represented by a set of chains of alphanumeric characters. The text is put together according to a grammar assigned to the language. A grammar is both the mathematical system used to define a formal language, and a set of rules to determine the syntactic validity of a specific sentence. Only context-free grammars are considered here, because higher programming languages (C, C++, Java, etc.) are described almost exclusively by context-free grammars. The following definitions apply:

Definition 1 *A context-free grammar is a 4-tuple* $G = (N,T,P,S)$, *where*

N *is the set of non-terminal symbols,*

T *is the set of terminal symbols, and*

P *is the relation* $N \times (N \cup T)^*$.

S *is a special non-terminal symbol, also called start symbol.*

Definition 2 *An element* $(A, \beta) \in P$ *is called* production *and is abbreviated* $A \rightarrow \beta$.

A production (or derivation) is thus a replacement rule for a non-terminal symbol A, such that this non-terminal

symbol is replaced by a string (chain) β composed of (non-terminal and) terminal symbols.

When there exist several productions (A, β_1), (A, β_2), ..., $(A, \beta_n) \in P$, for this we write:

$$\begin{aligned} A &\rightarrow \beta_1 \\ &\mid \beta_2 \\ &\vdots \\ &\mid \beta_n \end{aligned}$$

Definition 3 *The set $P_{A0} \subseteq P$ of all possible derivations for a non-terminal symbol A_0*

$$P_{A0} = \{(A,\beta) \in P : A = A_0\}$$

is also called the set of the alternatives *for A_0. The decision to use a specific production $(A_0,\beta_k) \in P_{A0}$, $(k = 1,2, \ldots, n)$ is called* selection.

Example 1 *All arithmetic expressions in the variables a with the operations + and * as well as arbitrarily nested brackets (,) are a formal language generated by the grammar*

$$G = (\{E,T,F\},\{\mathtt{a},+,*,(,)\},P,E).$$

P consists of the following productions:

$$\begin{aligned} E &\rightarrow E+T \\ &\mid T \\ T &\rightarrow T*F \\ &\mid F \\ F &\rightarrow (E) \\ &\mid \mathtt{a} \end{aligned}$$

a * **a** + (**a** * **a** + **a**) * **a**, *for example, is a valid sentence.*

In generating a sentence, further strings of non-terminal and, where appropriate, terminal symbols are derived by application of suitable productions to the non-terminal symbols until ultimately the string consists only of terminal symbols. In the example given above, the valid sentence consists only of the terminal symbols **a**, +, *, (,).

6.2 Principle of Syntax-Directed Coding

Information always flows into a text being generated when during its construction *decisions* are made. Decisions are to be made when a particular one of several possible source symbols is to be chosen or, in the case of a syntactically structured source, out of several applicable productions $A \rightarrow \beta_1$, $A \rightarrow \beta_2$, ... precisely one production actually is applied (selection, cf. Definition 3).

SDC encodes, instead of the string of terminal symbols, the selections in the order in which they were made during the construction of this string (the text).

Definition 4 $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ *is called SDC alphabet, with*

$$n = \max_A \{|P_A|\}$$

All the elements in each set of alternative P_a are numbered (arbitrarily) with SDC symbols $\sigma_1 \in \Sigma$:

Definition 5 $z_A : P_A \rightarrow \Sigma$ *is an (arbitrary) injective mapping.*

The concrete definition of z_A can in general be dependent on P_A.

As an illustration of the invention the following exemplary embodiments are described in detail and shown in Figures 1 to 6, as follows.

6.3 Brief Description of the Drawings

Figure 1 is a block diagram of an SDC transmission system

Figure 2 shows alternatives of the grammar *G* and selections to which SDC symbols have been attributed

Figure 3 is a parse tree of the sentence **a** * **a** + (**a** * **a** + **a**) * **a** with a resulting linear SDC symbol sequence 1212221211212222222

Figure 4 is a flow diagram of the stack machine;

Figure 5 is an example of an SDC transmission system with receiver-side source-text output

Figure 6 is an example of an SDC transmission system with executable Java applet or application as output

The functioning of a transmission system with SDC is illustrated in Figure 1. In principle transmission and storage are equivalent with respect to the way the coding works.

The source delivers a syntactically structured stream of symbols (source program). An elementary component of the SDC encoder is a parser, the task of which is to retrieve from the initially linear sequence of source symbols the grammatical structure of the sentence (the source program). This grammatical structure is represented in general by a parse tree, which in turn serves as input to the encoder. The encoder encodes the parse tree to generate a linear sequence of SDC symbols, such that at each moment the current SDC symbol is output and transmitted in dependence on the current selection at that moment. The SDC decoder first reconstructs the parse tree from the linear sequence of SDC symbols and then, by traversing the parse tree, reproduces the sequence of terminal symbols sent out from the source.

6.4 Encoding

The elementary task of the parser comprises the reconstruction of the *selections* made by the source during construction of the sentence. A parsing step thus amounts to determining the specific production $(A, \beta) \in P_A$ that the source had selected for constructing the text at this point. As stated in Definition 5, the encoder assigns to the selection thus determined the associated SDC symbol $\sigma_k = z_A((A, \beta))$ and attributes σ_k to the corresponding node in the parse tree.

By traversing the parse tree, ultimately the linear sequence of SDC symbols is generated and output. In the examples given here, a depth-first algorithm is always

used for this purpose. The process of encoding will now be explained in greater detail with reference to an exemplary parse tree and the underlying grammar from Example 1.

Example 2 *The context-free grammar of Example 1 is assumed. All alternatives, i.e. all possibilities for deriving each non-terminal symbol, are shown graphically in Figure 2. Because the greatest set of alternatives comprises precisely two elements, the SDC alphabet* $\Sigma = \{1, 2\}$ *suffices. To each selection of all quantities of alternatives an SDC symbol is attributed. For example, the selection of the specific production* $T \rightarrow T * F$ *is assigned the SDC symbol 1.*

A valid sentence in this grammar is thus, for example: $a * a + (a * a + a) * a$. *The parse tree for this sentence and the generation of the SDC symbol sequence by depth-first traversing are shown in Figure 3.*

6.5 SDC with arithmetic encoding

The method that has just been described is efficient only if

1. $|P_A| = const \ \forall \ A$, and
2. $P((A_1 \ \beta_i)|A) = const \ \forall \ A, \beta_i$,

where $P((A_1 \ \beta_i)|\alpha)$ is the conditional probability of making the selection that generates β_1 from the set of alternatives P_A.

This deficiency can be alleviated by arithmetic encoding [3] of the SDC alphabet Σ as stated in Claims 6 and 7. The probability distributions of the SDC symbols used for arithmetic encoding should be adapted in an adapter according to Claim 10 in which case the adaptation must be carried out separately for each set of alternatives. Instead of the linear sequence of SDC symbols, the bit stream generated by the arithmetic encoder is transmitted.

6.6 Decoding

Decoding is the inverse of the encoding process. The decoder formats the SDC code back into the original sentence (strings of terminal symbols). It operates according to the procedure described in Claims 1, 3 and 4, 5 and is implemented by a stack machine [4] according to Claims 8 and 9.

A flow diagram for a method according to these claims in shown in Figure 4.

To explain operation of the stack machine the following abbreviations are employed:

β_k String of terminal and non-terminal symbols

σ SDC symbol

V Terminal or non-terminal symbol

S Start symbol (non-terminal symbol)

The decoding thus proceeds in the following steps (the step numbers corresponding to the reference numerals shown in Figure 4:-

1. The development of a valid sentence according to the grammar (a program) begins with the start symbol S. The start symbol S is placed on the stack.
2. The uppermost symbol V is read from the stack.
3. Check whether V is a non-terminal symbol.
4. From the linear input stream of SDC symbols the next SDC symbol σ is read.
5. In case V is a non-terminal symbol, the derivation with V is further developed. By way of σ a selection $(V, \beta_k) = z^{-1}(\sigma)$ is made for V from the set of alternatives of $V \rightarrow \beta_1 | \beta_2 | \ \ldots \ | \beta_n$. In accordance with the specific selection, V is then replaced by the string β_k.
6. The string β_k of symbols (terminal and/or non-terminal symbols) is placed on the stack.
7. If V is a terminal symbol, it is output.
8. Check whether the stack is empty.
9. The stack machine terminates when the stack is empty.

The strings of terminal symbols output successively, when linked together, finally yield the sentence originally encoded from the source.

7. Exemplary Embodiment: Preferred Form of Application

The two exemplary embodiments shown in Figures 5 and 6 differ merely in the form in which data are sent to the receiver, namely as an output of source text corresponding to the method according to Claim 3 or as a direct output of an executable Java applet or an executable Java application, corresponding to the method according to Claim 4.

A description in greater details of these elements and their operation in the embodiments shown in Figures 5 and 6 is as follows.

1. Java source

   A Java source 1 corresponds to an ASCII text based on the Java grammar, which consists of Java program text plus comments, blank spaces etc.

2. SDC encoder

   The SDC encoder 2 is composed of a compiler front end, comprising a scanner 3, a parser 4 and a symbol-table memory 6, together with a parse-tree processor 5, an adapter 8 and modules for compression and concatenation of the generated data.

3. Scanner

   The scanner 3 reads the source text, removes the text sites that are unimportant for the program such as documentation, comments, blank spaces etc. and sends (terminal) symbols to the parser 4.

4. Parser

The parser 4 interprets the (terminal) symbols from the scanner 3, tests whether the symbol sequences agree with the grammatical rules for Java and sets up the specific parse tree for this program.

5. Parse-tree processor

During the initialization of parse-tree processing all nodes of the parse tree are assigned SDC symbols specified for each type of node. The subsequent traversing of all nodes of the parse tree causes an SDC symbol to be output at each crossing of a node. These SDC symbols are sent to both the adapter 8 and the arithmetic coder 10. In addition, the parse-tree processor 5 transmits the current node type to the adapter 8.

6. Symbol-table memory

The symbol-table memory 6 contains the identifiers of the Java program text extracted by the parser 4.

7. Zip-compression device

The content of the symbol table 6 is compressed by a method according to Lempel-Ziv-Welch [2].

8. Adapter (transmitter-side)

During the process of initialization the adapter 8 enters initial probability distributions for each node type into the probability-distribution table 9. During processing of the SDC symbols received from the parse-tree processor 5, and with reference

to the current node type, at each step the probability distribution of all SDC symbols of the current node type is adapted and entered into the probability-distribution table 9. The adapter 8 simultaneously makes available to the arithmetic encoder 10 the probability distribution associated with the SDC symbol currently being processed.

9. Table of the probability distributions of the SDC symbols

For each node type, the table 9 contains a probability distribution of SDC symbols that is peculiar to that node type.

10. Arithmetic encoder

The arithmetic encoder 3 receives from the parse-tree processor 5 the next SDC symbol to be encoded and encodes it with reference to the probability distribution made available by the adapter 8. The output of the arithmetic encoder 3 is a binary data stream.

11. Data concatenation

The binary data stream from the arithmetic encoder 10 and the symbol table 6 compressed by the compressor 7 are combined to form a data packet.

12. HTTP server

On the HTTP server 12 the data packet created in the data concatenation 11 is deposited and made available to be called up from a mass storage device.

13. Intranet/Internet

14. HTTP client (Internet browser)
The browser 14 has the task of downloading the data packet from an HTTP server 12 by way of the intranet/Internet 13 and initiating the process of decoding by the SDC decoder 15.

15. SDC decoder
The SDC decoder 15 comprises substantially of the arithmetic decoder 17 and the stack machine 22. The result of its operation is the sequence of (terminal) symbols that correspond to the original Java source code 1 (cf. 3 and 4).

16. Data extractor
Here the binary data stream from the arithmetic encoder 10 and the symbol table compressed by the zip compressor 7 are extracted from the common data packet and sent on, the former to the arithmetic decoder 17 and the latter to the zip-decompressor 18.

17. Arithmetic decoder
The arithmetic decoder 3 adaptively decodes the binary data stream produced by data extraction 16, taking into account the probability distributions associated with the SDC symbols that have been made available by the receiver-side adapter 20, and at each decoding step transmits an SDC symbol to the stack machine 22.

18. Zip-decompressor

The symbol table from the coder 2, compressed by the Lempel-Ziv method 7 and received from the data extractor 16, is unpacked and entered into the symbol-table memory of the decoder 19.

19. Symbol-table memory of the decoder

20. Adapter (receiver-side)

The initialization of the receiver-side adapter 20 is analogous to that of the transmitter-side adapter 8. While decoding proceeds, at each decoding step the adapter receives an SDC symbol, adapts the probability distribution of the SDC symbol of the current node type and enters the adapted probability distributions into the probability-distribution table 21. At the same time, the receiver-side adapter 20 supplies to the arithmetic decoder 17 the probability distribution that is valid for the next decoding step. The selection of this distribution is determined by the node type determined by the stack machine 22 and expected in the next decoding step.

21. Table of probability distributions of the SDC symbols

For each node type the table contains an individual probability distribution of the SDC symbols of this node type.

22. Stack machine

The initialization of the stack machine 22, the processing of the stack memory 23 and the output of terminal symbols is represented in Figure 4 as a flow diagram. In addition the stack machine 22 transmits the following data to the adapter 20:

(a) the current node type, for adaptation of the probability distributions of the SDC symbols of this node type; and

(b) the next node type, the SDC-symbol probability distribution of which the adapter 20 must supply to the arithmetic decoder 17 in the next decoding step.

23. Stack memory

Stack memory 23 for the stack machine 22.

24. Java sink

Retrieved Java program text without comments, superfluous blank spaces etc.

25. Code generator

The code generator 25 corresponds to a compiler back end and generates byte code for the JVM 26 in dependence on the sequence of (terminal) symbols provided by the stack machine 22.

26. JVM

The virtual machine for Java programs (Java Virtual Machine).

References

[1] HUFFMAN, D.A.: *a Method for the Construction of Minimum-Redundancy Codes.* Proc. IRE, 40:1098-1101, 1952

[2] ZIV, JACOB and ABRAHAM LEMPEL: *A Universal Algorithm for Sequential Data Compression.* IEEE Trans. Inform. Theory, IT-23(3):337-343, May 1977

[3] WITTEN, IAN H., RADFORD M. NEAL and JOHN G. CLEARY: *Arithmetic Coding for Data Compression.* Commun. ACM, 30(6):520-540, June 1987

[4] AHO, ALFRED V., RAVI SETHI and JEFFREY D. ULLMAN: *Compilerbau.* Addison-Wesley Publishing Company, Bonn, 4th Edition, 1988

Key to reference characters

Reference characters in Figure 4

β_k String of terminal and non-terminal symbols

σ SDC symbol

V Terminal or non-terminal symbol

S Start symbol (non-terminal symbol)

1. The development of a valid sentence according to the grammar (a program) begins with the start symbol S. The start symbol S is placed on the stack.
2. The uppermost symbol V is read from the stack.
3. Check whether V is a non-terminal symbol.
4. From the linear input stream of SDC symbols the next SDC symbol σ is read.

5. If V is a non-terminal symbol, the derivation is continued with V. By means of σ a selection $(V, \beta_k) = z^{-1}(\sigma)$ from the set of alternatives of $V \rightarrow \beta_1 | \beta_2 | \ldots | \beta_n$ is determined for V. According to the particular selection made, V is replaced by the string β_k.
6. The string β_k of symbols (terminal and/or non-terminal symbols) is placed on the stack.
7. If V is a terminal symbol, it is output.
8. Test whether the stack is empty.
9. The stack machine terminates when the stack is empty.

Reference characters in Figures 5 and 6

1. Java source
2. SDC encoder
3. Scanner
4. Parser
5. Parse-tree processor
6. Symbol-table memory
7. Zip-compression device
8. Transmitter-side adapter
9. Table of probability distributions of the SDC symbols
10. Arithmetic encoder
11. Data concatenation
12. HTTP server
13. Intranet/Internet
14. HTTP client (Internet browser)
15. SDC decoder
16. Data extractor
17. Arithmetic decoder

18. Zip-decompressor
19. Symbol-table memory of the decoder
20. Adapter (receiver side)
21. Table of probability distributions of the SDC symbols
22. Stack machine
23. Stack memory
24. Java sink
25. Code generator
26. JVM

Abstract of the Disclosure

A method and apparatus for its implementation are proposed that make it possible to eliminate redundancy in digitally represented texts with syntactic structure. The data reduction is brought about here by parsing the message according to the rules of the underlying syntax and subsequent entropy encoding of the decisions made to select the rules. If the text is a computer program, a stack machine can be used to generate from the compressed representation either the original text or a program that can be executed directly. Among the areas of application is the transmission of computer programs in networks with limited bandwidth.

27428/36596

For Filing Sec.371 National Stage Only
"Application With Literal Translation"
534 Rec'd PCT/PTO 28 JUL2000

PCT/DE99/00213

5/PRTS

COMPRESSION OF DATA WITH SYNTACTIC STRUCTURE

1. Introduction to the General Topic

The coding of source files has important technical applications in the area of transmission and storage of information. The desire to have a low data rate in the transmission of information, and a small space requirement in the storage of information, has given rise to the demand for a form of information coding that is as brief as possible, i.e. one with little or, ideally, no redundancy.

2. State of the Art

2.1. Forms of Redundancy

Redundancy in streams of source symbols takes two basic forms: it can appear as a differential frequency of occurrence of the elements in the set of source symbols, and in the form of statistical dependencies between temporally separated source symbols, which usually are caused by source-specific *formation laws* for the construction of a text as a sequence of source symbols.

3. Disadvantages of Known Methods

For the compression of messages from sources that suffer from redundancy and emit source symbols that are not uniformly distributed but are statistically independent, a procedure according to Huffman [1] is known. Being specially intended for this type of source, the

procedure enables maximal compression only under the extremely favorable condition that the statistical properties of the source are known or can be reliably estimated.

Whereas estimation of the probabilities of occurrence of individual source symbols, which is necessary for Huffman coding, can be done with sufficient accuracy even in the case of moderately long texts, the estimation of statistical dependencies between temporally separate source symbols as a rule requires extensive observation of the source, which goes far beyond what is feasible both technically (storage capacity) and in terms of observation time. For this reason the known or similar procedures always involve making assumptions about the underlying formation law or crucial characteristics thereof, and then of course are suitable only for sources with this formation law. This qualification also applies to the Lempel-Ziv (LZ) procedure for data compression [2], which has been known since the end of the 1970's. This procedure is suitable for the compression of arbitrary linear symbol sequences; that is, it can be applied with no knowledge of the statistical properties of the information source. The main areas of application today are the compression of pictorial data (GIF file format) and arbitrary files (archiving programs gnuzip, pkzip). In contrast to Huffman coding, this procedure utilizes statistical dependencies between temporally successive symbols, by taking into account repeatedly occurring subsequences of symbols (strings). Hence LZ compression is suitable only for sources, the formation law of which leads to the frequent occurrence of particular strings. The

redundancy introduced into a text by following grammatical rules during its construction is not accessible to the known compression procedures because of the recursive structure of grammatical rules, since these procedures merely employ symbol frequencies or repetitive chains of alphanumeric characters for compression.

4. Definition of the Object of the Invention

The object of the present invention is to provide a method and apparatus for implementing the method that enable a text consisting of a stream of alphanumeric characters, and is constructed according to grammatical rules, to be encoded in such a way as to remove the redundancy produced as a result of the limitation of all possible character sequences by the grammar.

This object is achieved by a method with the characteristics given in Claim 1 and two kinds of apparatus by means of which the characteristics given in Claims 8 and 9 are achieved.

5. Advantages of the Solution in Accordance with the Invention

With the solution in accordance with the invention, enabling data reduction in dependence on the programming-language syntax employed, it now becomes possible in an especially advantageous manner to remove, asymptotically almost completely, redundancy produced in texts during their construction because of the need to follow certain grammatical rules.

The invention has proved this by analytical calculation of the capacity of its data stream, encoded in a syntax-directed manner, using the example of a language with characteristics typical of programming languages (mathematical expressions, if-then-else construct, etc.). In this process it was shown that an appreciable amount of the redundancy that conventional methods (Lempel-Ziv, Huffman) are designed to address has not previously been eliminated by these methods.

The new method of syntax-directed coding, in contrast, is capable of reducing the amount of data by an additional *ca.* one-half in comparison, for instance, to a result obtained with the Lempel-Ziv compression procedure.

Thus the method in accordance with the invention, together with the apparatus specified for its implementation, is predestined for all applications that involve the transmission or storage of syntactically structured texts. Examples include the transmission of Java applets in the Internet or an intranet, the transmission or storage of Postscript files, the transmission and storage of MPEG4-encoded video data, and the use of higher protocols in communication. Especially for the transmission of programs, such as Java applets, syntax-directed coding (SDC) is particularly suitable because the parser is a component of both a compiler and an SDC coder, so that SDC can be organically integrated into the data flow from the writing to the running of the program.

6. Mode of operation of the invention

6.1 Syntactically structured source

The invention takes as a point of departure a formal language represented by a set of chains of alphanumeric characters. The text is put together according to a grammar assigned to the language. A grammar is both the mathematical system used to define a formal language, and a set of rules to determine the syntactic validity of a specific sentence. Only context-free grammars are considered here, because higher programming languages (C, C++, Java, etc.) are described almost exclusively by context-free grammars. The following definitions apply:

Definition 1 *A context-free grammar is a 4-tuple* $G = (N, T, P, S)$, *where*

N *is the set of non-terminal symbols,*

T *is the set of terminal symbols, and*

P *is the relation* $N \times (N \cup T)^*$.

S *is a special non-terminal symbol, also called start symbol.*

Definition 2 *An element* $(A, \beta) \in P$ *is called* production *and is abbreviated* $A \rightarrow \beta$.

A production (or derivation) is thus a replacement rule for a non-terminal symbol A, such that this non-terminal symbol is replaced by a string (chain) β composed of (non-terminal and) terminal symbols.

When there exist several productions (A, β_1), (A, β_2),

..., $(A, \beta_n) \in P$, for this we write:

$$A \rightarrow \beta_1 \mid \beta_2 \mid \beta_n$$

Definition 3 *The set $P_{A0} \subseteq P$ of all possible derivations for a non-terminal symbol A_0*

$$P_{A0} = \{(A,\beta) \in P : A = A_0\}$$

is also called the set of the alternatives *for A_0. The decision to use a specific production $(A_0,\beta_k) \in P_{A0}$, $(k = 1,2, \ldots, n)$ is called* selection.

Example 1 *All arithmetic expressions in the variables* a *with the operations* + *and* * *as well as arbitrarily nested brackets* (,) *are a formal language generated by the grammar*

$$G = (\{E,T,F\},\{a,+,*,(,)\},P,E).$$

P consists of the following productions:

$$E \rightarrow E+T \mid T$$

$$T \rightarrow T*F \mid F$$

$$F \rightarrow (E) \mid a$$

a * a + (a * a + a) * a, *for example, is a valid sentence.*

In generating a sentence, further strings of non-terminal and, where appropriate, terminal symbols are derived by application of suitable productions to the

non-terminal symbols until ultimately the string consists only of terminal symbols. In the example given above, the valid sentence consists only of the terminal symbols a, +, *, (,).

6.2 Principle of Syntax-Directed Coding

Information always flows into a text being generated when during its construction *decisions* are made. Decisions are to be made when a particular one of several possible source symbols is to be chosen or, in the case of a syntactically structured source, out of several applicable productions $A \rightarrow \beta_1$, $A \rightarrow \beta_2$, ... precisely one production actually is applied (selection, cf. Definition 3).

SDC encodes, instead of the string of terminal symbols, the selections in the order in which they were made during the construction of this string (the text).

Definition 4 $\Sigma = \{\sigma_1, \sigma_2, \ldots, \sigma_n\}$ *is called SDC alphabet, with*

$$n = \max_A \{|P_A|\}$$

All the elements in each set of alternative P_a are numbered (arbitrarily) with SDC symbols $\sigma_1 \in \Sigma$:

Definition 5 $z_A : P_A \rightarrow \Sigma$ *is an (arbitrary) injective mapping.*

The concrete definition of z_A can in general be dependent

on P_A.

As an illustration of the invention the following exemplary embodiments are described in detail and shown in Figures 1 to 6, as follows:

Figure 1 is a block diagram of an SDC transmission system

Figure 2 shows alternatives of the grammar *G* and selections to which SDC symbols have been attributed

Figure 3 is a parse tree of the sentence a * a + (a * a + a) * a with a resulting linear SDC symbol sequence 1212221211212222222

Figure 4 is a flow diagram of the stack machine

Figure 5 is an example of an SDC transmission system with receiver-side source-text output

Figure 6 is an example of an SDC transmission system with executable Java applet or application as output

The functioning of a transmission system with SDC is illustrated in Figure 1. In principle transmission and storage are equivalent with respect to the way the coding works.

The source delivers a syntactically structured stream of symbols (source program). An elementary component of the SDC encoder is a parser, the task of which is to

retrieve from the initially linear sequence of source symbols the grammatical structure of the sentence (the source program). This grammatical structure is represented in general by a parse tree, which in turn serves as input to the encoder. The encoder encodes the parse tree to generate a linear sequence of SDC symbols, such that at each moment the current SDC symbol is output and transmitted in dependence on the current selection at that moment. The SDC decoder first reconstructs the parse tree from the linear sequence of SDC symbols and then, by traversing the parse tree, reproduces the sequence of terminal symbols sent out from the source.

6.3 Encoding

The elementary task of the parser comprises the reconstruction of the *selections* made by the source during construction of the sentence. A parsing step thus amounts to determining the specific production $(A, \beta) \in P_A$ that the source had selected for constructing the text at this point. As stated in Definition 5, the encoder assigns to the selection thus determined the associated SDC symbol $\sigma_k = z_A((A, \beta))$ and attributes σ_k to the corresponding node in the parse tree.

By traversing the parse tree, ultimately the linear sequence of SDC symbols is generated and output. In the examples given here, a depth-first algorithm is always used for this purpose. The process of encoding will now be explained in greater detail with reference to an exemplary parse tree and the underlying grammar from

Example 1.

Example 2 *The context-free grammar of Example 1 is assumed. All alternatives, i.e. all possibilities for deriving each non-terminal symbol, are shown graphically in Figure 2. Because the greatest set of alternatives comprises precisely two elements, the SDC alphabet* $\Sigma = \{1, 2\}$ *suffices. To each selection of all quantities of alternatives an SDC symbol is attributed. For example, the selection of the specific production* $T \rightarrow T * F$ *is assigned the SDC symbol 1.*

A valid sentence in this grammar is thus, for example: a * a + (a * a + a) * a. *The parse tree for this sentence and the generation of the SDC symbol sequence by depth-first traversing are shown in Figure 3.*

6.4 SDC with arithmetic encoding

The method that has just been described is efficient only if

1. $|P_A| = const \ \forall \ A$, and
2. $P((A_1 \ \beta_1)|A) = const \ \forall \ A, \ \beta_i$,

where $P((A_1 \ \beta_1)|\alpha)$ is the conditional probability of making the selection that generates β_1 from the set of alternatives P_A.

This deficiency can be alleviated by arithmetic encoding [3] of the SDC alphabet Σ as stated in Claims 6 and 7. The probability distributions of the SDC symbols used

for arithmetic encoding should be adapted in an adapter according to Claim 10 in which case the adaptation must be carried out separately for each set of alternatives. Instead of the linear sequence of SDC symbols, the bit stream generated by the arithmetic encoder is transmitted.

6.5 Decoding

Decoding is the inverse of the encoding process. The decoder formats the SDC code back into the original sentence (strings of terminal symbols). It operates according to the procedure described in Claims 1, 3 and 4, 5 and is implemented by a stack machine [4] according to Claims 8 and 9.

A flow diagram for a method according to Claim 22 is given in Figure 4.

To explain operation of the stack machine the following abbreviations are employed:

β_k String of terminal and non-terminal symbols

σ SDC symbol

V Terminal or non-terminal symbol

S Start symbol (non-terminal symbol)

The decoding thus proceeds in the following steps (the numbers correspond to the reference numerals shown in Figure 4):

1. The development of a valid sentence according to the grammar (a program) begins with the start

symbol S. The start symbol S is placed on the stack.

2. The uppermost symbol V is read from the stack.
3. Check whether V is a non-terminal symbol.
4. From the linear input stream of SDC symbols the next SDC symbol σ is read.
5. In case V is a non-terminal symbol, the derivation with V is further developed. By way of σ a selection $(V, \beta_k) = z^{-1}(\sigma)$ is made for V from the set of alternatives of $V \rightarrow \beta_1 | \beta_2 | \ldots | \beta_n$. In accordance with the specific selection, V is then replaced by the string β_k.
6. The string β_k of symbols (terminal and/or non-terminal symbols) is placed on the stack.
7. If V is a terminal symbol, it is output.
8. Check whether the stack is empty.
9. The stack machine terminates when the stack is empty.

The strings of terminal symbols output successively, when linked together, finally yield the sentence originally encoded from the source.

7. Exemplary Embodiment: Preferred Form of Application

The two exemplary embodiments shown in Figures 5 and 6 differ merely in the form in which data are sent to the receiver (output of source text corresponding to the method according to Claim 3 versus direct output of an executable Java applet or an executable Java application, corresponding to the method according to

Claim 4).

1. Java source
   A Java source corresponds to an ASCII text based on the Java grammar, which consists of Java program text plus comments, blank spaces etc.

2. SDC encoder
   The SDC encoder is composed of a compiler front end ((3),(4) and(6)),a parse-tree processor (5), an adapter (8) and modules for compression and concatenation of the generated data.

3. Scanner
   The scanner reads the source text, removes the text sites that are unimportant for the program such as documentation, comments, blank spaces etc. and sends (terminal) symbols to the parser (4).

4. Parser
   The parser interprets the (terminal) symbols from the scanner (3), tests whether the symbol sequences agree with the grammatical rules for Java and sets up the specific parse tree for this program.

5. Parse-tree processing
   During the initialization of parse-tree processing all nodes of the parse tree are assigned SDC symbols specified for each type of node. The subsequent traversing of all nodes of the parse tree causes an SDC symbol to be output at each crossing of a node. These SDC symbols are sent to both the adapter (8) and the arithmetic coder (10). In addition, the parse-tree processor transmits the

current node type to the adapter (8).

6. Symbol-table memory
The symbol-table memory contains the identifiers of the Java program text extracted by the parser (4).

7. Zip-compression device
The content of the symbol table (6) is compressed by a method according to Lempel-Ziv-Welch [2].

8. Adapter (transmitter-side)
During the process of initialization the adapter enters initial probability distributions for each node type into the probability-distribution table (9). During processing of the SDC symbols received from the parse-tree processor (5), and with reference to the current node type, at each step the probability distribution of all SDC symbols of the current node type is adapted and entered into the probability-distribution table (9). The adapter 8 simultaneously makes available to the arithmetic encoder (10) the probability distribution associated with the SDC symbol currently being processed.

9. Table of the probability distributions of the SDC symbols
For each node type, the table contains a probability distribution of SDC symbols that is peculiar to that node type.

10. Arithmetic encoder
The arithmetic encoder (3) receives from the parse-

tree processor (5) the next SDC symbol to be encoded and encodes it with reference to the probability distribution made available by the adapter (8). The output of the arithmetic encoder is a binary data stream.

11. Data concatenation
The binary data stream from the arithmetic encoder (10) and the symbol table (6) compressed by the compressor (7) are combined to form a data packet.

12. HTTP server
On the HTTP server the data packet created in the data concatenation (11) is deposited and made available to be called up from a mass storage device.

13. Intranet/Internet

14. HTTP client (Internet browser)
The browser has the task of downloading the data packet from an HTTP server (12) by way of the intranet/Internet (13) and initiating the process of decoding by the SDC decoder (15).

15. SDC decoder
The SDC decoder comprises substantially of the arithmetic decoder (17) and the stack machine (22). The result of its operation is the sequence of (terminal) symbols that correspond to the original Java source code (1) (cf. 3 and 4).

16. Data extractor

Here the binary data stream from the arithmetic encoder (10) and the symbol table compressed by the zip compressor (7) are extracted from the common data packet and sent on, the former to the arithmetic decoder (17) and the latter to the zip-decompressor (18).

17. Arithmetic decoder
The arithmetic decoder [3] adaptively decodes the binary data stream produced by data extraction (16), taking into account the probability distributions associated with the SDC symbols that have been made available by the receiver-side adapter (20), and at each decoding step transmits an SDC symbol to the stack machine (22).

18. Zip-decompressor
The symbol table from the coder (6), compressed by the Lempel-Ziv method (7) and received from the data extractor (16), is unpacked and entered into the symbol-table memory of the decoder (19).

19. Symbol-table memory of the decoder

20. Adapter (receiver-side)
The initialization of the receiver-side adapter 20 is analogous to that of the transmitter-side adapter (8). While decoding proceeds, at each decoding step the adapter receives an SDC symbol, adapts the probability distribution of the SDC symbol of the current node type and enters the adapted probability distributions into the probability-distribution table (21). At the same

time, the receiver-side adapter supplies to the arithmetic decoder the probability distribution that is valid for the next decoding step. The selection of this distribution is determined by the node type determined by the stack machine (22) and expected in the next decoding step.

21. Table of probability distributions of the SDC symbols
For each node type the table contains an individual probability distribution of the SDC symbols of this node type.

22. Stack machine
The initialization of the stack machine, the processing of the stack memory and the output of terminal symbols is represented in Figure 4 as a flow diagram. In addition the stack machine transmits the following data to the adapter(20):

(a) the current node type, for adaptation of the probability distributions of the SDC symbols of this node type

(b) the next node type, the SDC-symbol probability distribution of which the adapter (20) must supply to the arithmetic decoder (17) in the next decoding step.

23. Stack memory
Stack memory for the stack machine.

24. Java sink

Retrieved Java program text without comments, superfluous blank spaces etc.

25. Code generator

The code generator corresponds to a compiler back end and generates byte code for the JVM (26) in dependence on the sequence of (terminal) symbols provided by the stack machine (22).

26. JVM

The virtual machine for Java programs (Java Virtual Machine).

References

[1] Huffman, D.A.: *a Method for the Construction of Minimum-Redundancy Codes*. Proc. IRE, 40:1098-1101, 1952.

[2] Ziv, Jacob and Abraham Lempel: *A Universal Algorithm for Sequential Data Compression*. IEEE Trans. Inform. Theory, IT-23(3):337-343, May 1977.

[3] Witten, Ian H., Radford M. Neal and John G. Cleary: *Arithmetic Coding for Data Compression*. Commun. ACM, 30(6):520-540, June 1987.

[4] Aho, Alfred V., Ravi Sethi and Jeffrey D. Ullman: *Compilerbau*. Addison-Wesley Publishing Company, Bonn, 4th Edition, 1988.

Key to reference characters

Reference characters in Figure 4

β_k String of terminal and non-terminal symbols

σ SDC symbol

V Terminal or non-terminal symbol

S Start symbol (non-terminal symbol)

1. The development of a valid sentence according to the grammar (a program) begins with the start symbol S. The start symbol S is placed on the stack.
2. The uppermost symbol V is read from the stack.
3. Check whether V is a non-terminal symbol.
4. From the linear input stream of SDC symbols the next SDC symbol σ is read.
5. If V is a non-terminal symbol, the derivation is continued with V. By means of σ a selection $(V, \beta_k) = z^{-1}(\sigma)$ from the set of alternatives of $V \rightarrow \beta_1 | \beta_2 | \ldots | \beta_n$ is determined for V. According to the particular selection made, V is replaced by the string β_k.
6. The string β_k of symbols (terminal and/or non-terminal symbols) is placed on the stack.
7. If V is a terminal symbol, it is output.
8. Test whether the stack is empty.
9. The stack machine terminates when the stack is empty.

Reference characters in Figures 5 and 6

1. Java source
2. SDC encoder
3. Scanner
4. Parser

5. Parse-tree processor
6. Symbol-table memory
7. Zip-compression device
8. Transmitter-side adapter
9. Table of probability distributions of the SDC symbols
10. Arithmetic encoder
11. Data concatenation
12. HTTP server
13. Intranet/Internet
14. HTTP client (Internet browser)
15. SDC decoder
16. Data extractor
17. Arithmetic decoder
18. Zip-decompressor
19. Symbol-table memory of the decoder
20. Adapter (receiver side)
21. Table of probability distributions of the SDC symbols
22. Stack machine
23. Stack memory
24. Java sink
25. Code generator
26. JVM

Claims

1. Method for the compressed transmission and/or storage of a text represented by digital data, the structure of which is defined by a grammar, characterized in that

   a) prior to storage and/or transmission the text is converted into a linear sequence of symbols that specify the successive application of grammatical rules to form the text, and

   (b) to decode the text after the storage and/or transmission, for every symbol thus produced by syntax-directed coding (SDC) the grammatical rule corresponding to that particular SDC symbol is performed and particular output data are generated by this rule.

2. Method according to Claim 1, characterized in that to generate the SDC symbols

   (a) by parsing the text a parse tree corresponding to the consecutive applications of the grammatical rules is generated,

   (b) to each node in this parse tree there is attributed an SDC symbol, which out of all the rules permitted by the grammar at this site unambiguously identifies the particular rule that is actually used to generate the original text, and

(c) the SDC symbols are concatenated, according to a fixed order of traversing all nodes of the parse tree, to form a linear sequence of SDC symbols.

3. Method according to Claim 1, characterized in that for decoding

(a) the uppermost entry in a stack memory is replaced according to the grammar production determined by the SDC symbol that has been input,

(b) parts of the grammatical rules that cannot yet be completely processed are deposited in a stack memory, and

(c) parts of the production for which substitution is complete are output as part of the original text.

4. Method according to Claim 1, characterized in that for decoding

(a) the uppermost entry in a stack memory is replaced according to the grammar production determined by the SDC symbol that has been input,

(b) parts of the grammatical rules that cannot yet be completely processed are deposited in a stack memory, and

(c) parts of the production for which substitution is complete are processed immediately to form part of an executable program for a real processor or a virtual machine.

5. Method according to Claim 3 or 4, characterized in that a stack machine

(a) is initialized by depositing a specific start symbol (non-terminal symbol) into the empty stack memory,

(b) reads the uppermost symbol from the stack memory,

(c) checks whether the read symbol is a terminal or a non-terminal symbol,

(d) if it is a terminal symbol, outputs the symbol and, depending on whether additional symbols are present in the stack memory, either continues with 5b or terminates if the stack memory is empty, or

(e) if it is a non-terminal symbol, reads the next SDC symbol from the input stream,

(f) depending on what SDC symbol has been read, selects precisely one alternative (chain of terminal and/or non-terminal symbols) out of the set of alternatively applicable replacement rules (productions) that are valid

for the non-terminal symbol currently being processed, and

(g) places this chain of terminal and/or non-terminal symbols into the stack memory and continues with 5b.

6. Method according to Claim 2, characterized in that

(a) to each node in the parse tree there is attributed an SDC symbol and the probability distribution of all the SDC symbols possible in this node,

(b) the linear sequence of SDC symbols is subjected to entropy encoding in conformity with the associated probability distributions,

(c) the entropy decoding is carried out with a probability distribution of SDC symbols identical to that used for the entropy encoding.

7. Method according to Claim 6, characterized in that

(a) the probability distribution of the rules that can be applied in a node, starting from an initial distribution, is adapted at each appearance of an SDC symbol in such a way that the probability of the SDC symbol that appears is increased and the probability of all other

symbols is correspondingly reduced,

(b) the currently valid distribution of occurrence probabilities is assigned to the SDC symbols of the associated node type,

(c) the probability distribution of all SDC symbols in a current node, together with the SDC symbol to be encoded, forms a model for an arithmetic encoding,

(d) during decoding the end of the text is recognized by the fact that the stack memory is empty, and

(e) an End-Of-Message (EOM) symbol required for arithmetic coding is eliminated.

8. Apparatus for the method according to Claim 1, characterized by an encoder comprising

(a) a scanner to transform text comprising a sequence of readable characters into a sequence of terminal symbols,

(b) a parser to find grammatical rules, the successive application of which was originally used to generate the sequence of terminal symbols,

(c) a mapper, which unambiguously associates syntax-directed symbols with the rules identified by the parser and outputs these

symbols in a fixed sequence,

and a decoder comprising

(a) a stack machine which, according to the uppermost symbol in the stack memory and, where appropriate, the adjacent SDC symbol, outputs the already fixed terminal symbol, or deposits the sequence of terminal and/or non-terminal symbols associated with the current symbol into the stack memory, and

(b) a lexicon that replaces the terminal symbols by chains of readable alphanumeric characters.

9. Apparatus for the method according to Claim 3, characterized by an encoder comprising

a) a scanner to transform a program that is present in a source text or in a form derived from the source text by a preprocessor, into a sequence of terminal symbols,

(b) a parser to find the grammatical rules, the successive application of which was originally used to generate the sequence of terminal symbols, and

(c) a mapper, which unambiguously associates syntax-directed symbols with the rules identified by the parser and outputs these symbols in a fixed sequence,

and a decoder comprising

(a) a stack machine which, according to the uppermost symbol in the stack memory and, where appropriate, the adjacent SDC symbol, outputs the already fixed terminal symbol, or deposits the sequence of terminal and/or non-terminal symbols associated with the current symbol into the stack memory, and

(b) a code generator that generates from the sequence of terminal symbols executable machine code, or intermediate code to be executed on a virtual machine.

10. Apparatus for the method according to Claim 7, comprising

(a) on a transmitter side

i. a table that contains the probability distributions of the SDC symbols for each node type, the contents of which are established at initialization with fixed initial probability distributions for each node type,

ii. an adapter that updates the probability distribution of the SDC symbols for the node type valid at any given moment with reference to the existing probability distribution, the SDC symbol to be encoded and the current node type, and enters this new probability distribution

into the table, and

iii. an arithmetic coder that encodes each SDC symbol to be encoded with the currently valid probability distribution supplied by the adapter;

(b) and on a receiver side

i. a table that contains the probability distributions of the SDC symbols for each node type, the contents of which are established at initialization with fixed initial probability distributions for each node type,

ii. an adapter that updates the probability distribution of the SDC symbols for the node type valid at any given moment, as established by the stack machine, with reference to the existing probability distribution, the SDC symbol to be encoded and the current node type, and enters this new probability distribution into the table,

iii. an arithmetic decoder which, with reference to the currently valid probability distribution of the current node type supplied by the adapter, decodes the next SDC symbol and sends it to the stack machine for further processing.

Abstract

A method and apparatus for its implementation are proposed that make it possible to eliminate redundancy in digitally represented texts with syntactic structure.

The data reduction is brought about here by parsing the message according to the rules of the underlying syntax and subsequent entropy encoding of the decisions made to select the rules. If the text is a computer program, a stack machine can be used to generate from the compressed representation either the original text or a program that can be executed directly.

Among the areas of application is the transmission of computer programs in networks with limited bandwidth.

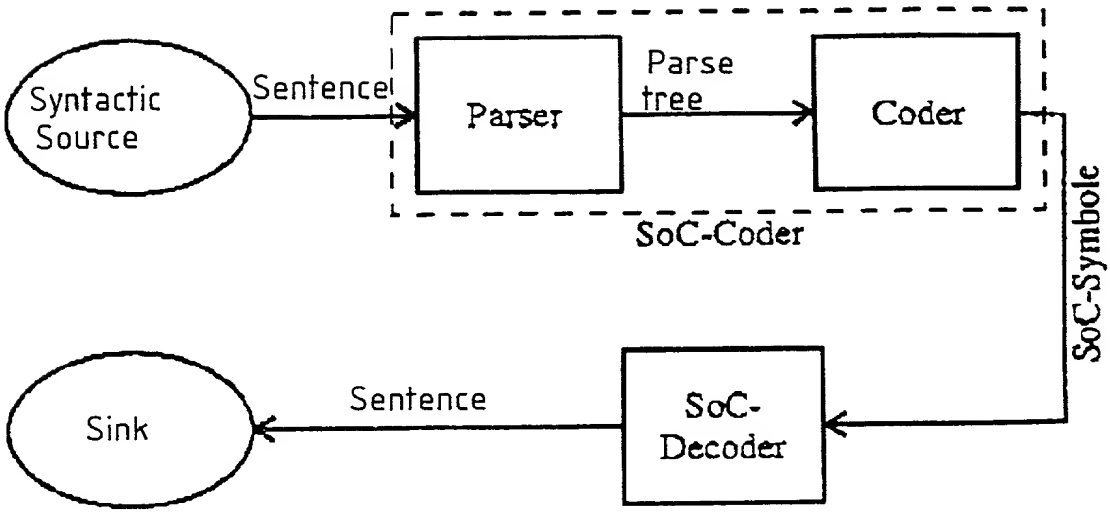


Figure 1

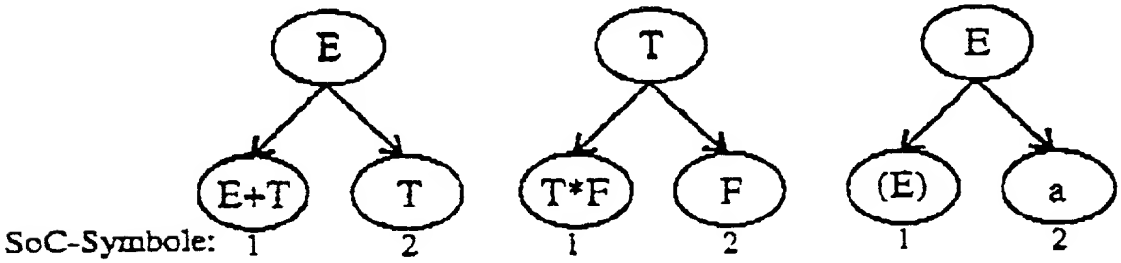


Figure 2

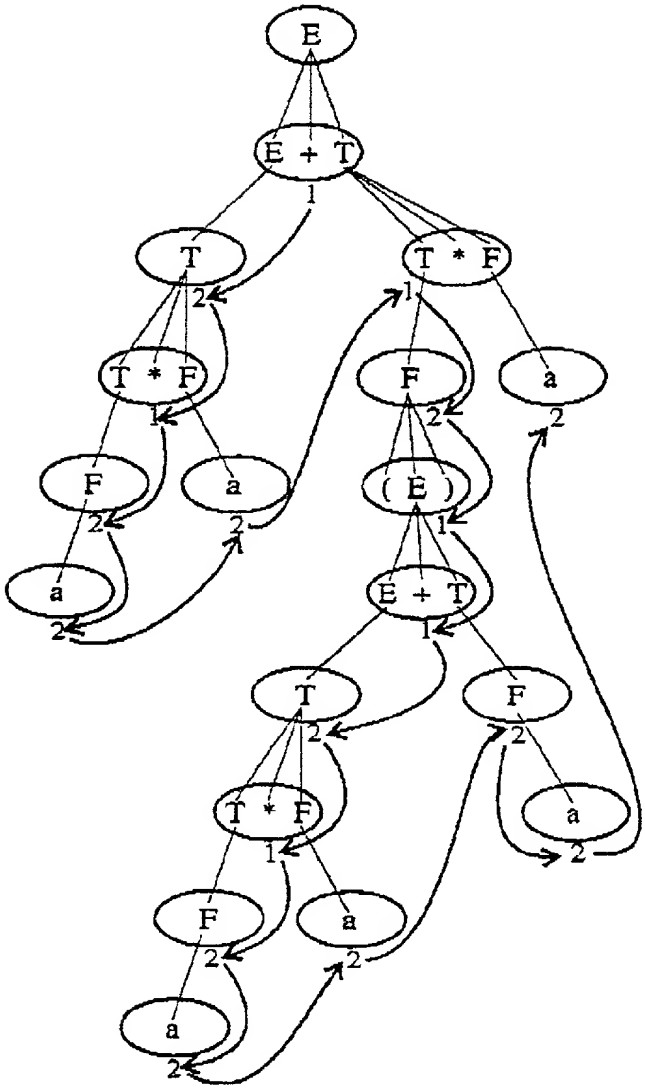


Figure 3

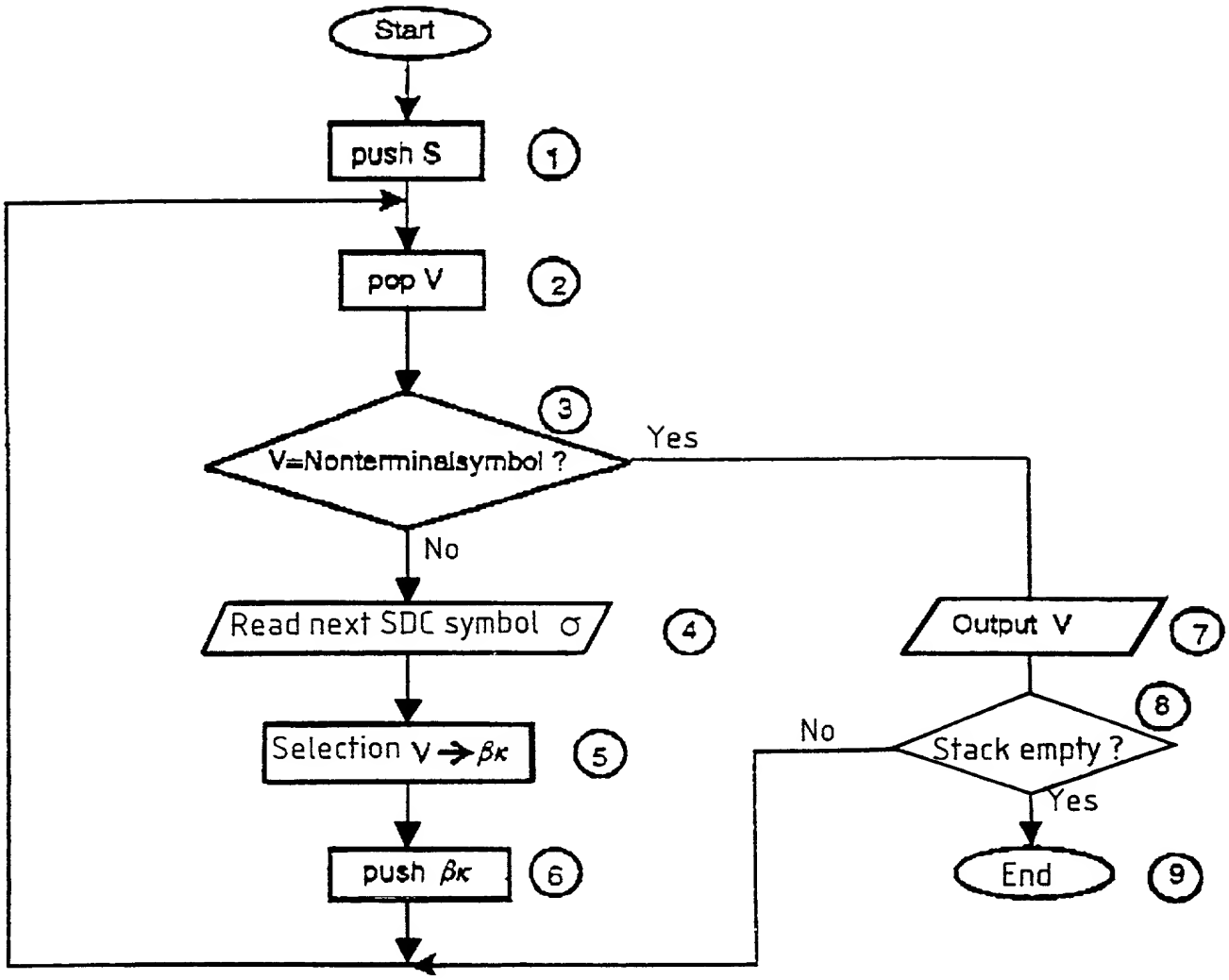

Start
push S
1
pop V
2
3
V=Nonterminalsymbol ?
Yes
No
Read next SDC symbol σ
4
Selection V → βκ
5
push βκ
6
Output V
7
8
Stack empty ?
No
Yes
End
9

Figure 4

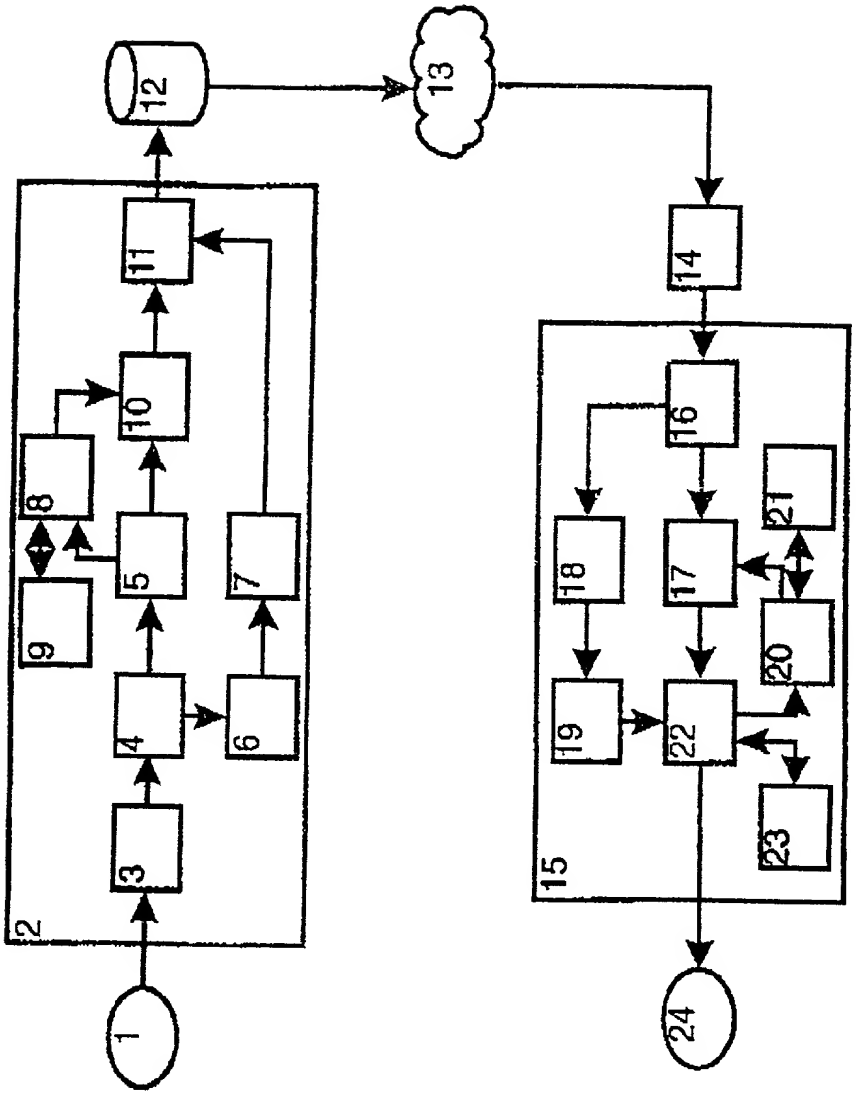


Figure 5

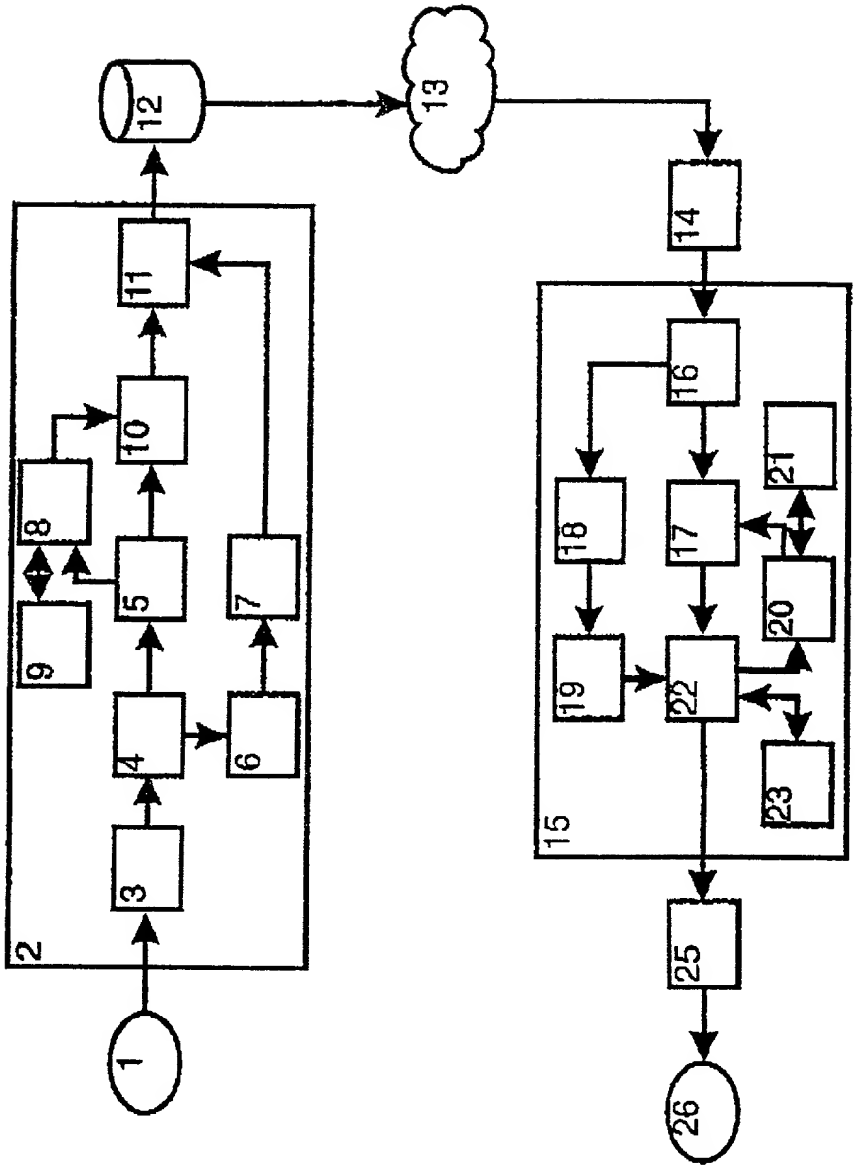


Figure 6

DECLARATION FOR PATENT APPLICATION AND POWER OF ATTORNEY

As a below named inventor, I hereby declare that my residence, post office address and citizenship are as stated below next to my name; I believe that I am the original, first and sole inventor (if only one name is listed below) or an original, first and joint inventor (if plural names are listed below) of the subject matter which is claimed and for which a patent is sought on the invention entitled "**COMPRESSION OF DATA WITH SYNTACTIC STRUCTURE,**" the specification of which (check one): ☐ is attached hereto; ☒ was filed on July 28, 2000 as Application Serial No. 09/601,167 and was amended on _____ (if applicable); ☐ was filed as PCT International Application No. _____ on _____ and was amended under Article 19 on _____ (if applicable). I hereby state that I have reviewed and understand the contents of the above-identified specification, including the claims, as amended by any amendment(s) referred to above. I acknowledge the duty to disclose to the Patent and Trademark Office all information known to me to be material to patentability as defined in 37 C.F.R. §1.56.

I hereby claim foreign priority benefits under 35 U.S.C. §119 of any foreign application(s) for patent or inventor's certificate or of any PCT international application(s) designating at least one country other than the United States of America listed below and have also identified below any foreign application(s) for patent or inventor's certificate or any PCT international application(s) designating at least one country other than the United States of America filed by me on the same subject matter having a filing date before that of the application(s) of which priority is claimed:

			Priority Claimed	
198 03 845.3	Germany	31 January 1998	☒ Yes	☐ No
(Application Serial Number)	(Country)	(Day/Month/Year Filed)		
			☐ Yes	☐ No
(Application Serial Number)	(Country)	(Day/Month/Year Filed)		

I hereby claim the benefit under 35 U.S.C. §119(e) of any United States provisional application(s) listed below:

(Application Serial Number)	(Day/Month/Year Filed)
(Application Serial Number)	(Day/Month/Year Filed)

I hereby claim the benefit under 35 U.S.C. §120 of any United States application(s) or PCT international application(s) designating the United States of America listed below and, insofar as the subject matter of each of the claims of this application is not disclosed in the prior application(s) in the manner provided by the first paragraph of 35 U.S.C. §112, I acknowledge the duty to disclose to the Office all information known to me to be material to patentability as defined in 37 C.F.R. §1.56 which occurred between the filing date of the prior application(s) and the national or PCT international filing date of this application:

PCT/DE99/00213	28 January 1999	Pending
(Application Serial Number)	(Day/Month/Year Filed)	(Status-Patented, Pending or Abandoned)
(Application Serial Number)	(Day/Month/Year Filed)	(Status-Patented, Pending or Abandoned)

I hereby declare that all statements made herein of my own knowledge are true and that all statements made on information and belief are believed to be true; and further that these statements were made with the knowledge that willful false statements and the like so made are punishable by fine or imprisonment, or both, under 18 U.S.C. §1001 and that such willful false statements may jeopardize the validity of the application or any patent issued thereon.

POWER OF ATTORNEY: I hereby appoint as my attorneys, with full powers of substitution and revocation, to prosecute this application and transact all business in the Patent and Trademark Office connected therewith: From: 0823

27

Alvin D. Shulman (19,412)	Richard H. Anderson (26,526)	Jeffrey S. Sharp (31,879)	David W. Clough (36,107)
Allen H. Gerstein (22,218)	Patrick D. Ertel (26,877)	Martin J. Hirsch (32,237)	Richard A. Brandon (37,051)
Nate F. Scarpelli (22,320)	James P. Zeller (28,491)	James J Napoli (32,361)	James A. Flight (37,622)
Edward M. O'Toole (22,477)	William E. McCracken (30,195)	Richard M. La Barge (32,254)	Roger A. Heppermann (37,641)
Michael F. Borun (25,447)	Richard A Schnurr (30,890)	Li-Hsien Rin-Laures, M.D (33,547)	David A. Gass (38,153)
Trevor B. Joike (25,542)	Anthony Nimmo (30,920)	Douglass C Hochstetler (33,710)	Gregory C. Mayer (38,238)
Carl E. Moore, Jr. (26,487)	Christine A. Dudzik (31,245)	Robert M. Gerstein (34,824)	

Send correspondence to: Nate F. Scarpelli

FIRM NAME	PHONE NO.	STREET	CITY & STATE	ZIP CODE
Marshall, O'Toole, Gerstein, Murray & Borun	312-474-6300	6300 Sears Tower 233 South Wacker Drive	Chicago, Illinois	60606-6402

1-00

Full Name of First or Sole Inventor	Citizenship
PETER ECK	German
Residence Address - Street	Post Office Address - Street
Wangenerstraße 75	Wangenerstraße 75
City (Zip)	City (Zip)
Starnberg 82319	Starnberg 82319
State or Country	State or Country
Federal Republic of Germany DEX	Federal Republic of Germany
Date ☒ 28-Aug.-2000	Signature ☒ Peter Eck

200

Second Joint Inventor, if any	Citizenship
ROLF MATZNER	German
Residence Address - Street	Post Office Address - Street
Josef-Frankl-Strasse 9a	Josef-Frankl-Strasse 9a
City (Zip)	City (Zip)
80995 Munich DEX	80995 Munich
State or Country	State or Country
Federal Republic of Germany	Federal Republic of Germany
Date ☒ 28-AUG-2000	Signature ☒

300

Third Joint Inventor, if any	Citizenship
CHANGSONG XIE	Chinese
Residence Address - Street	Post Office Address - Street
Maximilian-Kolbe-Allee 15	Maximilian-Kolbe-Allee 15
City (Zip)	City (Zip)
81739 Munich DE X	81739 Munich
State or Country	State or Country
Federal Republic of Germany	Federal Republic of Germany
Date ☒ 28-Aug-2000	Signature ☒ Changsong Xie

Fourth Joint Inventor, if any	Citizenship
Residence Address - Street	Post Office Address - Street
City (Zip)	City (Zip)
State or Country	State or Country
Date ☒	Signature ☒

APPLICABLE RULES AND STATUTES

37 CFR 1.56. DUTY OF DISCLOSURE - INFORMATION MATERIAL TO PATENTABILITY (Applicable Portion)

(a) A patent by its very nature is affected with a public interest. The public interest is best served, and the most effective patent examination occurs when, at the time an application is being examined, the Office is aware of and evaluates the teachings of all information material to patentability. Each individual associated with the filing and prosecution of a patent application has a duty of candor and good faith in dealing with the Office, which includes a duty to disclose to the Office all information known to that individual to be material to patentability as defined in this section. The duty to disclose information exists with respect to each pending claim until the claim is canceled or withdrawn from consideration, or the application becomes abandoned. Information material to the patentability of a claim that is canceled or withdrawn from consideration need not be submitted if the information is not material to the patentability of any claim remaining under consideration in the application. There is no duty to submit information which is not material to the patentability of any existing claim. The duty to disclose all information known to be material to patentability is deemed to be satisfied if all information known to be material to patentability of any claim issued in a patent was cited by the Office or submitted to the Office in the manner prescribed by §§ 1.97(b)-(d) and 1.98. However, no patent will be granted on an application in connection with which fraud on the Office was practiced or attempted or the duty of disclosure was violated through bad faith or intentional misconduct. The Office encourages applicants to carefully examine:

(1) prior art cited in search reports of a foreign patent office in a counterpart application, and
(2) the closest information over which individuals associated with the filing or prosecution of a patent application believe any pending claim patentability defines, to make sure that any material information contained therein is disclosed to the Office.

Information relating to the following factual situations enumerated in 35 USC 102 and 103 may be considered material under 37 CFR 1.56(a).

35 U.S.C. 102. CONDITIONS FOR PATENTABILITY: NOVELTY AND LOSS OF RIGHT TO PATENT

A person shall be entitled to a patent unless --

(a) the invention was known or used by others in this country, or patented or described in a printed publication in this or a foreign country, before the invention thereof by the applicant for patent, or

(b) the invention was patented or described in a printed publication in this or a foreign country or in public use or on sale in this country, more than one year prior to the date of the application for patent in the United States, or

(c) he has abandoned the invention, or

(d) the invention was first patented or caused to be patented, or was the subject of an inventor's certificate, by the applicant or his legal representatives or assigns in a foreign country prior to the date of the application for patent in this country on an application for patent or inventor's certificate filed more than twelve months before the filing of the application in the United States, or

(e) the invention was described in a patent granted on an application for patent by another filed in the United States before the invention thereof by the applicant for patent, or on an international application by another who has fulfilled the requirements of paragraph (1), (2), and (4) of section 371(c) of this title before the invention thereof by the applicant for patent, or

(f) he did not himself invent the subject matter sought to be patented, or

(g) before the applicant's invention thereof the invention was made in this country by another who had not abandoned, suppressed, or concealed it. In determining priority of invention there shall be considered not only the respective dates of conception and reduction to practice of the invention, but also the reasonable diligence of one who was first to conceive and last to reduce to practice, from a time prior to conception by the other.

35 U.S.C. 103. CONDITIONS FOR PATENTABILITY; NON-OBVIOUS SUBJECT MATTER (Applicable Portion)

A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

Subject matter developed by another person, which qualifies as prior art only under subsection (f) or (g) of section 102 of this title, shall not preclude patentability under this section where the subject matter and the claimed invention were, at the time the invention was made, owned by the same person or subject to an obligation of assignment to the same person.

35 U.S.C. 112. SPECIFICATION (Applicable Portion)

The specification shall contain a written description of the invention, and of the manner and process of making and using it, in such full, clear, concise, and exact terms as to enable any person skilled in the art to which it pertains, or with which it is most nearly connected, to make and use the same, and shall set forth the best mode contemplated by the inventor of carrying out his invention.